Amendments to the Claims

This listing of claims will replace all prior versions and listings of claims in the application. Applicants have submitted a new complete claim set showing marked up claims with insertions indicated by underlining and deletions indicated by strikeouts and/or double bracketing.

Listing of Claims:

1. (Canceled)

2. (Previously Presented) The method of Claim 3 wherein said demand-loadable components are initially provided in one of:

(a) a memory within said computer;

(b) a location external of said computer.

3. (Previously Presented) A method of providing a dynamically configurable operating system on a computer having a working memory, comprising:

providing demand-loadable operating system components initially stored outside of said working memory, each operating system component having an entry point comprising a constructor for an object, and

providing a Namespace in said working memory which provides in said working memory access to ones of said operating system components as they become needed by applications running in said computer, said Namespace managing demand-loading and unloading of said operating system components in said working memory.

4. (Canceled)

5. (Previously Presented) The method of Claim 3 further comprising providing applications in said working memory which rely on said Namespace to furnish access to ones of said operating system components in said working memory as they become needed by ones of said applications.

6. (Previously Presented) The method of Claim 5 wherein each operating system component comprises an object, and wherein the step of providing each demand-loadable component comprises:

providing an IUnknown interface in the object having the following methods:

(a) add reference for incrementing a count of the number of applications requiring the object;

(b) release reference for decrementing a count of the number of applications requiring the object;

wherein said Namespace is responsive to said count in said managing of said demand-loading and unloading.

7. (Previously Presented) The method of Claim 5 wherein each operating system component comprises an object, and wherein the step of providing a demand-loadable operating system component comprises:

providing an IUnknown interface in the object having a QueryInterface method of providing access to the methods of the object to an application invoking QueryInterface.

8. (Original) The method of Claim 5 wherein said object is a COM object.

9. (Previously Presented) A method of operating a computer having a working memory wherein applications and objects may be loaded during run time, comprising:

providing a Namespace in said computer;

running an application in said computer;

said application calling a demand-loadable object by causing the name of said object to be presented to said Namespace, wherein said object is an operating system component;

in response to being presented with the name of said object, said Namespace returning to said application an IUnknown pointer of said object;

upon return of said IUnknown pointer of said object, said application using said IUknown pointer to call a QueryInterface method of said object and request a pointer to a desired interface;

said QueryInterface method returning said desired interface, whereby said application can invoke a desired method through said interface.

10. (Original) The method of Claim 9, wherein said computer comprises a loader, and wherein said Namespace, responds to being presented with the name of said object in that said Namespace:

determines whether the name of said object is currently registered in said Namespace, and, if so, carries out the step of returning said pointer;

if said name is not currently registered, causes said loader to load said object into said working memory and registers said name in said Namespace, and then carries out the step of returning said pointer.

11. (Original) The method of Claim 10 wherein said object has a constructor and an entry point, and wherein the loading of said object comprises:

said loader invoking said constructor;

said constructor finding said entry point of said object and calling an executable at said entry point;

said executable causing space in said working memory to be allocated for a VTable, an Interface and an Implementation of said object and producing a pointer to said memory space, said pointer comprising said IUnknown pointer.

12. (Original) The method of Claim 11 further comprising:

loading said VTable, Interface and Implementation in the space in said working memory allocated therefore;

initializing the state of said object including said VTable and interface pointers.

13. (Previously Presented) A computer having a working memory and access to a storage memory, said computer comprising:

an application capable of being loaded in to said working memory and running in said computer;

at least one object initially stored in said non-working memory, said object comprising an operating system component;

a Namespace in said working memory;

said application being programmed to cause said one object to be identified to said Namespace whenever said application finds a need for said object during the running of said application;

said Namespace being programmed to:

(a) respond to said application identifying said one object by determining whether said one object is currently registered in Namespace, and if it is not registered, then,

(b) causing said one object to be loaded from said storage memory to said working memory and,

(c) registering said one object in said Namespace,

(d) upon said object being registered in said Namespace, returning to said application a pointer to said object.

14. (Original) The computer of Claim 13 wherein said object comprises a VTable, plural interfaces and corresponding methods thereof, and plural implementations, one of said interfaces comprising an IUknown interface including a QueryInterface method, and wherein said application finds a need for said one object because said application needs a particular one said interfaces of said one object, wherein:

said pointer to said object returned by said Namespace comprises an IUnknown pointer to said IUknown interface of said one object;

said application is programmed to use said IUnknown pointer to access said IUknown interface of said one object and to invoke the QueryInterface method thereof to access said particular one interface.

15. (Original) The computer of Claim 14 wherein said computer further comprises a loader in said working memory capable of loading objects from said storage memory to said working memory, said Namespace causing said one object to be loaded by causing said loader to load said one object.

16. (Original) The computer of Claim 15 wherein said object has a constructor and an entry point, and wherein said loader is programmed such that said loader:

invokes said constructor;

said constructor is programmed to find said entry point of said object and call an executable at said entry point;

said executable is programmed to cause space in said working memory to be allocated for a VTable, an Interface and an Implementation of said object and producing a pointer to said memory space, said pointer comprising said IUnknown pointer.

17. (Original) The computer of Claim 16 wherein:

said loader is further programmed to load said VTable, Interface and Implementation in the space in said working memory allocated therefore;

said loader is programmed to initialize the state of said object including said VTable and interface pointers.

18. (Previously Presented) A computer having a working memory and access to a storage memory, said storage memory holding at least one object, said computer comprising:

an application in said working memory, said application needing to access to said one object at a particular time during the running of said application, said object comprising an operating system component, and said application being programmed to cause a request for said one object to issue contemporaneously with said particular time;

a Namespace which is programmed to respond to a request from said application for said one object by loading said one object from said storage memory

into said working memory and then providing said application with a pointer to said one object.

19. (Original) The computer of Claim 18 wherein:

said application is programmed to notify said Namespace whenever it no longer needs access to said one object;

said Namespace is further programmed to permit said one object to be unloaded from said working memory when no longer needed.

20. (Original) The computer of Claim 18 wherein:

said application is programmed to notify said one object it no longer needs access to it, whereby said object notifies said Namespace said object is no longer needed;

said Namespace is further programmed to permit said one object to be unloaded from said working memory when no longer needed.

21. (Original) The computer of Claim 18 wherein said Namespace permits said one object to remain in working memory after being no longer needed by said application in order to permit other applications to access said one object.

22. (Canceled)

23. (Previously Presented) The system of Claim 24 wherein said storage memory comprises one of:

(a) a memory within said computer;

(b) a memory external of said computer.

(c) the output of another software component such as a compiler.

24. (Previously Presented) A system residing in a working memory of a computer and in a storage memory, said system comprising:

demand-loadable operating system components initially stored in said storage memory, each operating system component having an entry point comprising a constructor for an object, and

a Namespace in said working memory which provides in said working memory access to ones of said operating system components as they become needed by applications running in said computer, said Namespace managing demand-loading and unloading of said operating system components in said working memory.

25. (Canceled)

26. (Previously Presented) The system of Claim 24 further comprising applications in said working memory which rely on said Namespace to furnish access to ones of said operating system components in said working memory as they become needed by ones of said applications.

27. (Previously Presented) The system of Claim 26 wherein each operating system component operating system comprises an object, and the system further comprises:

an IUnknown interface in the object having the following methods:

(a) add reference for incrementing a count of the number of applications requiring the object;

(b) release reference for decrementing a count of the number of applications requiring the object;

wherein said Namespace is responsive to said count in said managing of said demand-loading and unloading.

28. (Previously Presented) The system of Claim 27 further comprising:

an IUnknown interface in the object having a QueryInterface method of providing access to the methods of the object to an application invoking QueryInterface.

29. (Original) The system of Claim 26 wherein said object is a COM object.

30. (Original) The system of Claim 27 further comprising a loader in said working memory, and wherein said Namespace is responsive to being presented with the name of one of said objects by:

returning a pointer to said object if said object is registered in said Namespace and returning to said application a pointer to said object;

if said name is not currently registered, causing said loader to load said object into said working memory.

31. (Original) The system of Claim 30 wherein said object has a constructor invoked by said loader, an entry point and an working memory space-allocating executable called by said constructor at said entry point.

32. (Original) The system of Claim 31 wherein said object comprises a VTable, an Interface and an Implementation in locations in said working memory allocated by said executable.

33. (Previously Presented) A method of operating a computer having a working memory wherein applications and objects may be loaded during run time, comprising:

providing a Namespace in the computer;

providing a kernel resident in the working memory at run time;

providing a loadable interprocess communication manager resident at link time outside of the working memory and dynamically loadable into the working memory at run time, the interprocess communication manager having an IUnknown pointer, a QueryInterface method and plural interfaces;

running an application in the computer;

the application presents to the Namespace the name of the interprocess communication manager;

in response to the Namespace being presented by the application with the name of the interprocess communication manager, the Namespace returning to the application the IUnknown pointer of the interprocess communication manager;

upon return of the IUnknown pointer, the application using the IUnknown pointer to call the QueryInterface method of the interprocess communication manager, and requesting through the QueryInterface method a pointer to a desired interface of the interprocess communication manager;

the QueryInterface method returning the desired interface, whereby the application can invoke a desired method through the interface.

34. (Previously Presented) The method of Claim 33, wherein the computer comprises a loader in the working memory, and wherein the Namespace, prior to providing the IUnknown pointer, performs the steps of:

determining whether the interprocess communication manager is currently registered in the Namespace;

if the interprocess communication manager is not currently registered, causing the loader to load the interprocess communication manager into the working memory and registering the interprocess communication manager in the Namespace.

35. (Previously Presented) A method of operating a computer having a working memory wherein applications and objects may be loaded during run time, the method comprising:

providing a Namespace in the computer;

providing a kernel resident in the working memory at run time;

providing a loadable virtual memory manager resident at link time outside of the memory and dynamically loadable into the working memory at run time upon demand of one of the application programs, the virtual memory manager having an IUnknown pointer, a QueryInterface method and plural interfaces;

running an application in said computer;

said application presenting the name of the virtual memory manager to the Namespace;

in response to the Namespace being presented by the application with the name of the virtual memory manager, the Namespace returning to the application the IUnknown pointer of the virtual memory manager;

upon return of the IUnknown pointer, the application using the IUnknown pointer to call the QueryInterface method of the virtual memory manager and to request

through the QueryInterface method a pointer to a desired one of the plural interfaces of the virtual memory manager;

the QueryInterface method returning the desired interface, whereby the application can invoke a desired method through the interface.

36. (Previously Presented) The method of Claim 35, wherein the computer comprises a loader in the working memory, and wherein the Namespace, prior to providing the IUnknown pointer, performs the steps of:

determining whether the name of the virtual memory manager is currently registered in the Namespace;

if the name is not currently registered, causing the loader to load the virtual memory manager into the working memory and registering the virtual memory manager in the Namespace.

37. (Previously Presented) The method of Claim 35 wherein the plural interfaces of the virtual memory manager comprise:

virtual memory space interface (VMSpace);

virtual memory map interface (VMMap);

virtual memory view interface (VMView).